A method to generate simplified Systemic Functional Parses from Dependency Parses

Eugeniu Costetchi
CRP Henri Tudor, Luxembourg, 29, J.F. Kennedy, 1855, Luxembourg
eugeniu.costetchi@tudor.lu

Abstract

Systemic Functional Linguistics provides a semiotic perspective on language. The text analysis described in Systemic Functional Linguistics (SFL) can be of critical value in real-world applications. But parsing with SFL grammars is computationally intensive task and parsers for this level of description to date have not been able to operate on unrestricted input. This paper describes a graph-based method to automatically generate simplified SFL mood and transitivity parses of English sentences from Stanford Dependency parses and a database providing transitivity categories for each verb.

1 Introduction

Broad coverage natural language components now exist for several levels of linguistic abstraction, ranging from tagging and stemming, through syntactic analyses, to semantic specifications. In general, the higher the degree of abstraction, the less accurate coverage becomes.

Transitivity descriptions[1] as developed within Systemic Functional Linguistics (SFL) offer a semantically-oriented decomposition of clauses that is still sufficiently closely tied to observable grammatical distinctions as to offer a powerful bridge for automatic analysis. Transitivity analyses, like those in Table 1, provide descriptions analogous to frame descriptions (Fillmore, 1985; Minsky, 1974) as found in FrameNet (Baker, Fillmore, & Lowe, 1998) or VerbNet (Kipper, Korhonen, Ryant, & Palmer, 2008) which are applied in Semantic Role Labelling tasks. CoNLL-2004/5(Carreras & Màrquez, 2005) shared tasks on SRL, revealed a major performance drop when when the test corpus differs from the training one. This can be due to the use of machine learning or due to annotation schemas. By contrast to VerbNet & FrameNet, the SFL transitivity descriptions enforce a further generalisation across the kinds of frame roles that can be used. This generalisation allows descriptions to be preserved when clauses are realised in different forms and also provides the basis for making a more robust connection to structural syntactic features of clauses.

Mood descriptions offer a functional syntactic decomposition of clauses that serves as a well-argued foundation for transitivity analysis (Halliday & Matthiessen, 2004).

SFL adopts a semiotic perspective on language and distinguishes different meaning-lines fused in the text. Therefore parsing in terms of mood and transitivity (e.g. Table 1) can be of tremendous value for Natural Language Understanding and Critical Discourse Analysis. Provided automatic mood and transitivity parsing can have applications well beyond those traditionally explored with automatic semantic and syntactic analysis.

<table>
<tr><td>example 1</td><td>the duke</td><td colspan="2">had</td><td>given</td><td>the teapot</td><td>to my aunt.</td></tr>
<tr><td rowspan="2">mood</td><td colspan="6">clause: [mood type: declarative; tense: past perfect simple; voice: active: polarity: positive]</td></tr>
<tr><td>subject</td><td colspan="2">predicate
finite | predicator</td><td colspan="2">complement</td><td>complement</td></tr>
<tr><td>transitivity</td><td>agent-carrier</td><td colspan="3">possessive process</td><td>possessed</td><td>beneficiary</td></tr>
<tr><td>example 2</td><td>the lion</td><td colspan="3">caught</td><td>the tourist</td><td>yesterday.</td></tr>
<tr><td rowspan="2">mood</td><td colspan="6">clause: [mood type: declarative; tense: past perfect simple; voice: active: polarity: positive]</td></tr>
<tr><td>subject</td><td colspan="2">predicator/finite</td><td colspan="2">complement</td><td>adjunct</td></tr>
<tr><td>transitivity</td><td>agent-carrier</td><td colspan="3">possessive process</td><td>affected-possessed</td><td>temporal location</td></tr>
</table>

Table 1 sample mood and transitivity analyses

Parsers for this level of description to date have not been able to operate on unrestricted input. To parse directly in terms of SFL is a computationally difficult task. However there have been successful attempts to produce SFL parses in two steps. This paper describes a method to automatically generate English sentences parses of mood and transitivity from Stanford Dependency parses (Marneffe, MacCartney, & Manning, 2006; Marneffe & Manning, 2008) and from the Process Type Database (Neale, 2002).

[1] Note that transitivity in SFL is a clause-level representation and not a verb property such as in traditional grammars.

Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013), pages 68–77,
Prague, August 27–30, 2013.

Typed dependency grammars (like SD) and systemic functional grammars are different analysis approaches. The typed dependency analysis results in word pairs bound by a syntactic relation. The systemic functional analysis results in constituents and feature structures. The constituents are text chunks labelled with their functional grammatical role in the clause, while feature structures are sets of attribute-value pairs representing properties of constituents.

The main issues addressed here are: how to determine the boundaries of constituents and their (mood/transitivity) roles in a clause and how to further determine their features based on constituent position and lexico-grammatical resources. The Stanford Dependency Parser (SDP) offers a suitable backbone for bootstrapping mood analysis. The defined grammatical roles are syntactically compatible with functional grammar and contribute to the solving constituency problem (see Section 4.3). For the second problem we employ pattern graphs corresponding to choices in systemic networks together with lexical-semantic resources such as the PTDB to further enrich the constituents with semantic information.

The Process Type Database (PTDB) is a dictionary-like database of verb lexical items, each of which is bound to list of verb senses and corresponding semantic frames that dictate the process type and participant roles selection.

In the remainder of this paper we briefly introduce key SFL concepts with focus on mood and transitivity along with simplified MOOD[2] and TRANSITIVITY systems in terms of which we currently consider parsing (Section 2). Then, in Section 3, are presented the main contributions in SFL parsing followed, in Section 4, by description of our graph-based parsing approach detailed with parsing method, computational implementations and resources that support it. In Section 5 we conclude on presented approach.

2 SFL preliminaries

In Systemic Functional Linguistics there are three lines of meaning expressed in any clause: *textual*, *interpersonal* and *experiential*. Textually, a clause acts as a message (or an information unit) that contributes to the creation of the discourse as a whole. Interpersonally, a clause is a unit of exchange between speaker and listener, and so serves social relations and speech functions enacted in a clause. Within SFL, however, speech acts are expressed by means of typical grammatical variations and expressions, thus maintaining a tighter link between the speech act and the grammatical realization. *Mood analysis* provides the framework to grasp and use these grammatical variations. Experientially, a clause is the representation of some "*process in ongoing human experience*"(Halliday & Matthiessen, 2004, p. 170) and is described through *transitivity*.

Systemic Functional Grammar approach to syntactic structure is to focus on systematizing the *choice possibilities* a speaker has for construing her utterances. Each choice shapes the grammatical realization and is accompanied by a range of semantic implications. These choices are then structured in hierarchical system networks so that early choices restrict latter ones.

There are two large variants of Systemic Functional Grammars: the Sydney Grammar proposed by Halliday (Halliday & Matthiessen, 2004), who originated SFL, and the Cardiff Grammar proposed by Fawcett (Fawcett, 2008), who, based on Sydney Grammar, has constructed an alternative account focusing more directly on syntactic generalizations.

In the present paper, the Cardiff Grammar is used for transitivity analysis because the PTDB is build according to it and, a simplified version of Sydney Grammar is used for mood analysis as described in the next section.

2.1 Mood constituency and MOOD systemic network

Mood constituency analysis in SFL supports the interpersonal perspective on language and resembles the analysis of Quirk (1985) or that of Fawcett (2008) where the clause is syntactically split into constituting elements. We will refer to it as *mood constituency*, because, in SFL, all analyses have their own specific way of splitting the clause into constituents. An example of such analysis is exhibited in Table 1. The following is a brief description of clause constituents and their functional roles in exchange (argument) structure.

The *Finite* is a part of the verbal group expressing the tense or modality. It either precedes the Predicator (introduced below) or is conflated with it in present and past simple tenses. The role of the Finite is to make the clause finite by anchoring it into the here and now, so to speak, bringing the clause into the context of the speech event. This is done either by reference to the time

[2] Capitalized notation refers to a SFG system network whereas non-capitalized terms refer to concept s.

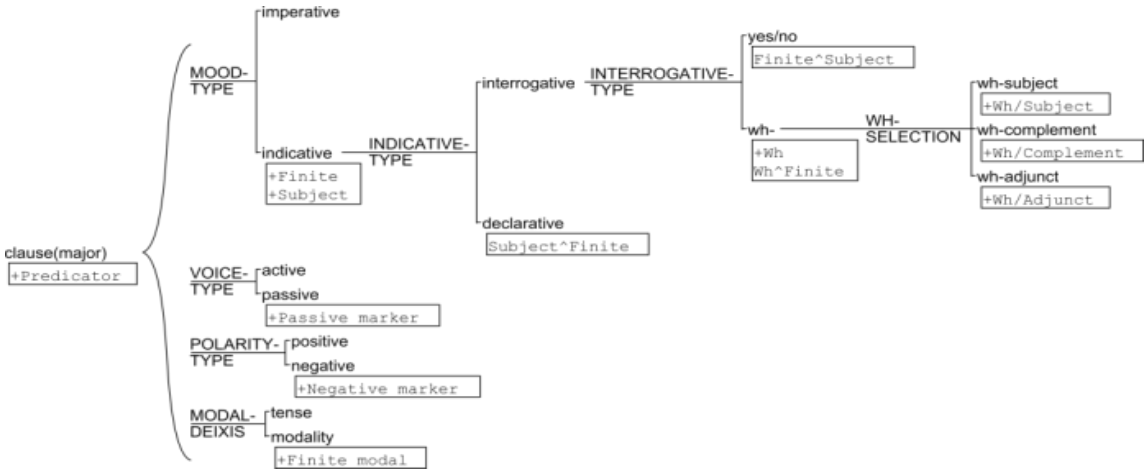

Figure 1 simplified MOOD system of Sydney Grammar

of speaking via tense or by reference to the judgment of the speaker via modality.

The *Subject* is the nominal group or a nominal clause that precedes the Predicator in a clause and it is something by reference to which the proposition can be affirmed or denied. It is considered to be "*modally responsible*" for the validity of what is being predicated (stated, commanded, questioned or offered) in the clause. Note that the predication is not interpreted as an experiential relation but as an interpersonal relation. Hence there is no interpretation in terms of truth values of a clause (because for e.g. offers, and commands cannot be attributed truth values).

The *Predicator* is the part of verbal group minus the finite constituent when they are not conflated. It specifies additional temporal and aspectual relations, voice and the process type (e.g. action, relation, mental process etc.) that is predicated about the Subject. It can contain one or more Main Verbs. The *Main Verb* is a non-auxiliary and non-modal verb at the end of the verbal group. If there is more than one Main Verb we say that it is a *complex clause*. To enforce the syntactic and functional analysis proposed in the Cardiff analysis methodology (Fawcett, 2008), the complex clauses need to be separated into individual clauses so that each comply with the "*one main verb per clause*" principle. Sentence division into clauses is explained in Section 2.5.

The *Complement* is a part of the clause that follows the Predicator and has the potential of becoming a Subject, i.e. it can become an axis of the argument. Usually it is a nominal group and rarely a prepositional phrase. For example in passive clauses the agent easily loses the preposition "*by*" to become Subject.

Complements correspond to "*objects*" in traditional grammars.

The *Adjuncts* are the last type of clause constituent. They do not have the potential of becoming a Subject; therefore arguments cannot be constructed around adjunct elements. They are realized by adverbial, nominal and prepositional groups.

The system of MOOD used in this paper (Figure 1) is a simplified version of Sydney

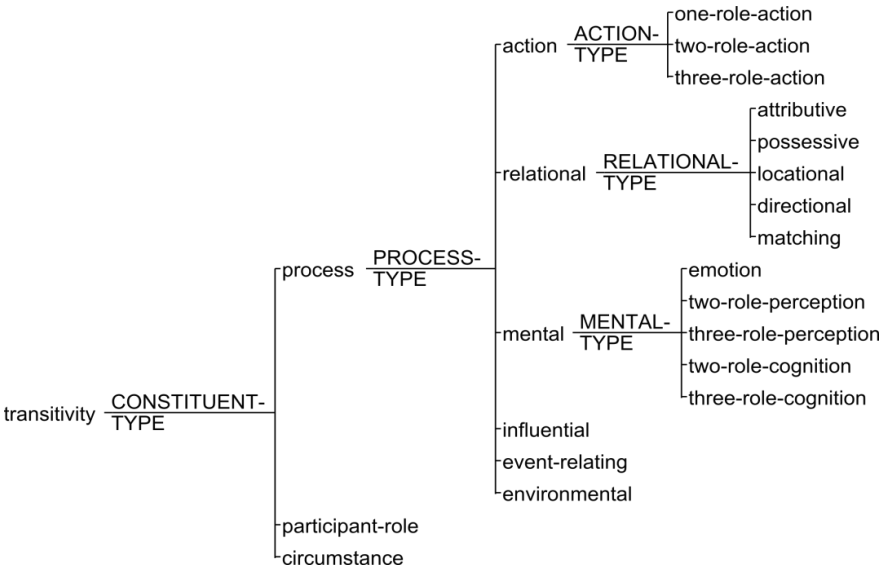

Figure 2 TRANSITIVITY system of Cardiff Grammar

Grammar mood. It focuses four features[3]: *mood type, voice type, clause polarity, and modal/temporal deixis*. These are clause-level features which are determined either by: (1) constituent presence, (2) constituent order, or (3) lexical items within a constituent. In section 4.3 is explained how to generate the mood structure and its features from the dependency graph.

2.2 Transitivity constituency and TRANSITIVITY systemic network

TRANSITIVITY (Figure 2) defines the *process types*, *participant roles* that correspond to each process type and *circumstances* that can occur in the language (English in this case). These are functional units of a configuration whose syntactic counterpart is the clause.

The *Process* is the central element of a configuration. Each *process type* (classification in Figure 2) provides its own model or schema for construing a particular domain of experience by defining a configuration of *participant roles* for that particular process type. The Process is filled by Finite and Predicator constituents but the Main Verb dictates systemic selection of the Process Type.

Participants are filled by Subject and Complement constituents and their roles are selected by the configuration schema. A configuration can have from one to three Participants just as a clause has a Subject up to two Complements. The vast majority of Processes require two Participants whereas only a small number of processes ask for one or three Participants.

The last unit type in a configuration is the *circumstance*. It introduces additional information about the configuration such as time, space, cause, manner, etc. Circumstances are filled by Adjunct constituents and are optional units in a configuration. The clause is syntactically valid if adjuncts are removed whereas if a Subject, Predicate or Complement is missing the clause changes its meaning or becomes syntactically invalid. The same holds for a configuration; if a participant or process is removed then it becomes another configuration or invalid.

One might argue that in "*John behaved well*", if we remove or substitute the adjunct "*well*", then the meaning of the entire clause is modified. The *Manner* is treated as circumstance in Sydney Grammar but in Cardiff grammar, it has been given a participant role. Since we are bond to the latter, syntactically manner is still an adjunct but semantically it becomes a participant role.

Due to space limitation, the detailed process type, participant role or circumstances classification are not covered further in the current paper. They are treated with great detail by Halliday and Matthiessen (2004), Neale (2002) and Fawcett (2009).

Seldom, a clause can be interpreted as corresponding to more than one configuration type which implies different participant role and process type selections. This principle, enounced by Halliday, is called *systemic indeterminacy* (Halliday & Matthiessen, 2004, p. 173) and applies to all systems but especially to TRANSITIVITY.

2.3 The Process Type Database

The Process Type Database (Neale, 2002) is the key resource in the automatisation of transitivity analysis because the selection of the process type during transitivity analysis is a semantically driven operation. PTDB provides information on what possible process types and participants can correspond to a particular verb.

The PTDB is a dictionary-like dataset of verb lexical items, each of them, bound to an exhaustive list of verb senses and the corresponding Process Configuration for each sense. It is the result of Neale's work (2002) on improving the TRANSITIVITY system of the Cardiff Grammar. She systematizes according to the Cardiff Grammar over 5400 senses (and process configurations) for over 2750 verbs. A small example is presented in *Table 2*. Each verb sense has its own Process Configuration and can coincide or differ from the Process Configurations of other verb senses.

verb form	*informal meaning*	*configuration*
calculate	work out by mathematics (commission will then calculate the number of casted votes)	cognition: Ag-Cog + Ph
	plan (newspaper articles were calculated to sway reader's opinions)	two role action: Ag + Cre
catch	run after and seize (a leopard unable to catch its normal prey)	possessive: Ag-Ca + Af-Pos
	(did you catch a cold?)	possessive: Af-Ca + Pos
catch (up with)	reach (Simon tried to catch up with others)	two role action: Ag + Ra

Table 2 sample PTDB entries (simplified)

[3] Feature values are further determined by their own sub-systems.

2.4 The interplay between mood and transitivity – the case of prepositional groups

There are cases in mood analysis when deciding the unit type is impossible by relying solely on syntactic analysis (including typed dependency analysis). Prominent cases are the *prepositional phrases*. These can fill both a Complement and an Adjunct role. For mood analysis this implies that the same syntactic unit can fill a Complement and an Adjunct, while for transitivity analysis, it implies that the same syntactic unit can fill a Participant or a Circumstance.

(1) *John goes home through London.*
(2) *John is building a house for Bob.*
(3) *her teardrop shines like a diamond.*
(4) *John is building a house for ten years now.*
(5) *John goes to London by fast train.*

In examples (1) and (2) the prepositional phrases "*through London*"[4] and "*for Bob*" are Complements and Participants (Path and Beneficiary roles) while in examples (3), (4) and (5), "*like a diamond*", "*for ten years now*" and "*by fast train*" are Adjuncts and Circumstances (of comparison, temporal duration and manner-means).

prep	*role*[5]	*Sydney grammar*	*Cardiff grammar*
by	Ag	material: actor; mental: phenomenon; relational: token	action: actor; mental (emotive): phenomenon; relational: token
to	Ben	material: recipient; verbal: receiver	action: client / receiver[6]
to	Dest	material: location / place	action: destination
for	Ben	material: client	action: receiver
as	Attr	relational: attribute	relational (attributive): attribute
on, in	Ra	material: scope; verbal: verbiage; material: locational / place	action: range / destination

Table 3 Prepositions introducing participants

To solve this problem of undetermined role allocation there are two complementary solutions. The first one is to mark the every prepositional phrase as Complement and as Adjunct. This just postpones the decision of selecting the right unit type, however.

The second solution is to decide based on the preposition and potential process type as specified in the PTDB. Most of prepositions introduce only circumstances and only a few prepositions can introduce participants as well. And when they do, it is for only specific process types. *Table 3* we present prepositions known to introduce participants for process types. This table is an extension of the one from (Halliday & Matthiessen, 2004, p. 278) and contains translations to Cardiff Grammar counterparts.

2.5 Sentence partition into clauses

Dependency Graphs (will be introduced in Section 4.2) are graphs of a whole sentence whereas transitivity analysis is at individual clause-level. This implies that DG need to be split into individual clauses before transitivity analysis. We propose to detect and delimit clauses during the mood analysis. For some commonly occurring situations we propose treatments aligned with Fawcett's (2008, 2009) methodology as follows.

When the clauses are connected by a conjunction and have their own subject/objects then the conjunction is the clause border marker.

(6) *The lion chased the tourist but she escaped alive.*
(6a) *The lion[Ag-Ca] chased[Pr] the tourist[Af-Pos]*
(6b) *she[Ag] escaped[Pr] alive[Ra]*

When the predicators are conjoined and share subject and/or objects then each predicator will form a new clause and borrow the subject/objects from the other clause.

(7) *The lion chased and caught the tourist.*
(7a) *the lion[Ag-Ca] chased[Pr] the tourist[Af-Pos]*
(7b) *the lion[Ag-Ca] caught[Pr] the tourist[Af-Pos]*

In the case of mental, influential and event relating processes (classification in Figure 2) the predicates are often complex. Verbs in these classes are known as control and raising verbs (Haegeman, 1991) where a superordinate controls subordinate non-finite verb and binds its participants (Subject/Complement).

In order to comply with "*one main verb per clause*" principle, each Main Verb of the complex clause becomes a governor of a distinct clause. The subordinate verb with all of its dependent nodes is assigned to a placeholder. The superordinate verb receives the placeholder as Complement with the role of Phenomena. If the subject is missing in the subordinate clause then it is copied from the superordinate one.

(8) *The lion wanted/began to chase the tourist.*
(8a) *the lion[Cog] wanted/began[Pr] X[Phen]*

[4] In Sydney Grammar it is a circumstance for a material process. However, in Cardiff Grammar for Directional and Locative Processes some circumstances are treated as participants therefore they are Complements (Fawcett, 2009).
[5] General functions defined in Sydney Grammar: Ag(Agent), Ben(Beneficiary), Dest(Destination), Attr(Attribute), Ra(Range), etc.
[6] Beneficiary and Client are not directly specified in Cardiff system. This role is identified as Destination in two and three role actions. The test distinguishing between beneficiary and destination is checking whether the participant is animate or non-animate.

(8b) *X= the lion[Ag-Ca] to chase[Pr] the tourist[Af-Pos]*

The meaning of complex clause decomposition can be expressed with an equivalent rephrasing by inserting "*something that is*" between the Main Verbs, as in example (9).

(9) *The lion wanted/began something that is to chase the tourist.*

3 Literature review

Most of the parsing attempts in SFL dealt with the Nigel grammar (Matthiessen, 1985), which is a large and complex natural language generation (NLG) grammar. One of the early attempts was done by Kasper (1988). He recompiles[7] the Nigel grammar as feature structures employing *Functional Unification Grammar* (FUG) (Martin Kay, 1985) which is a well-established and a formally understood representation. Kasper used phrase-structure trees which served as backbones to which were mapped systemic feature choices.

O'Donnell use a different approach to recompiling the Nigel grammar which allowed him to parse text directly without appeal to the phrase-structure backbone that Kasper had required (O'Donnell, 1993, 1994). However he could not parse with the entire Nigel grammar because of the sheer size of the grammar and its inherent complexity introduced by multiple parallel classifications (Bateman, 2008). O'Donnell (O'Donnell, 2005) subsequently, in UAM Parser, decided, for pragmatic reasons, to return to a syntactic backbone and restrict the grammar so that functionally only the Mood structure of clauses is accounted for.

In a very different style of approach, Honnibal and Curran (2005) constructed a parser to convert Penn Treebank into a corresponding SFGBank. This managed to provide a good conversion from parse trees into systemic functional representation covering sentence mood and thematic constituency (the third kind of analysis in SFL which has been mentioned in Section 2). Transitivity was not been covered because of its inherently semantic nature.

More recently, O'Donnell (2012) in UAM Corpus Tool, created a parser that uses Stanford Parser(Klein & Manning, 2003) output as a backbone, which then is transformed into mood parse and then further derives the Sydney Grammar transitivity parse. He uses a mood backbone and enriches this with semantic features that are derived based on lexical choices and structural patterns.

Our approach is aligned with Honnibal's and O'Donnell's work with respect to using mood constituency as a backbone and enriching it with syntactic and semantic features. When approaching transitivity, O'Donnell provides the possible process types that a verb can have by employing a large lexicon where each word has syntactic and semantic features. The approach described here differs both in terms of the lexical resource and parsing method used. We employ PTDB, which provides entire configurations (frames) for each verb sense and the parsing method is a graph-based pattern matching.

4 The parsing method

In this section implementations are proposed and their capacities described, as well as methods that perform mood and transitivity parsing. The Stanford Dependency Schema proposed in (Marneffe et al., 2006) and re-motivated in (Marneffe & Manning, 2008) constitutes the departing point of our current approach in building a Mood Constituency Graph (MCG). MCG is the structure reflecting mood analysis and serves as the backbone for performing transitivity analysis via Graph Matching operations. Our method involves three types of graph structures: (1) *Dependency Graphs*, (2) *Mood Constituency Graphs* and (3) *Pattern Graphs*. We now introduce the specifics of a generic graph structure and the operations that these graphs support and then we present the parsing algorithms.

4.1 The graphs and operations over them

Graphs are defined as usual as a data structure consisting of a finite set of directed *edges* connecting *node* entities. The nodes, however, are not atomic data but *Feature Structures* (Carpenter, 1992), whereas the edges are triples (x,y,f) where *x* and *y* are nodes being connected and *f* is the feature structure of the edge. A generic *feature structure* (FS) is a set of attribute-value pairs where the value can be of an atomic or a complex data-type such as list, dictionary or feature structure.

The literature on mood and transitivity analysis specifies a range of methods for detecting and selecting a particular feature (Fawcett, 2008, 2009; Halliday & Matthiessen, 2004). In order to support those methodological specifications the

[7] Recompilation is employed to adopt a resource for application needs. Nigel grammar was initially created for NL generation. That grammar structure is not applicable for the parsing task.

graphs need to allow a number of operations: (1) *querying* over nodes and edges, (2) *graph matching*, (3) *pattern matching* and (4) *pattern-based node extraction*.

Querying over the node or edge FS return nodes or edges that comply with the constraints of the query. For example one can ask for all nodes that contain an "*NP*" part of speech or all node pairs connected by "*det*" relation.

Graph matching enables answering questions of whether a graph is identical to a sub-graph of the second one. This is the *graph isomorphism problem*, and is known to be NP-complete. However, the available algorithm (Cordella, Foggia, Sansone, & Vento, 2004) nevertheless performs this task very quickly when the graphs addressed are of limited size. In our case the graphs are of (English) sentences composed in average of 15-20 words. This lies well within the limits of practical computability.

An extension of graph matching is the *pattern matching operation*. A *graph pattern* (GP) is a graph whose feature structures can either be under or over specified. In the case of underspecified FS, the attributes and/or their values can be omitted down to an empty feature structure. In the case of over specified FS, the values are a list of possible values for an attribute.

For example, Figure 3 depicts a GP for detecting present perfect continuous tense. The slash ("*/*") symbol stands for part of speech attribute, "at" ("@") stands for the lexeme attribute while square brackets ("*[,]*") indicate a list of values that are accepted for a match. Note that this pattern is underspecified for most attribute-value pairs and over specified for one edge indicating two acceptable edge types ("*[aux, auxpass]*") and for one node POS ("*[vbz, vbp]*").

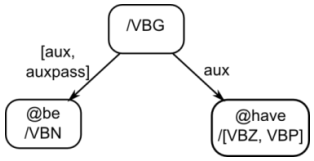

Figure 3 sample GP

The last operation is *pattern-based node extraction*. The purpose of the operation is returning nodes that have been marked in GP for extraction in the case of GP match. The matched nodes are returned together with the values of extraction markers in GP. An *extraction marker* is simply another attribute-value pair in the node's FS. This gives the possibility to assign new functional-semantic features to nodes, such as participant roles during transitivity parsing.

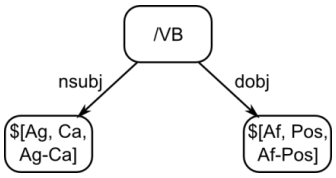

Figure 4 sample GP with marked nodes

For example, Figure 4 represents a GP used for transitivity analysis, where the dollar sign ("*$*") notation stands for an extraction marker. This means that whenever a verb is encountered that has a noun subject ("*nsubj*") and a direct object ("*dobj*"), then the subject node can receive agent, carrier or agent-carrier roles ("*[Ag,Ca,Ag-Ca]*"), while the object node can be attributed with affected, possessed or affected-possessed roles ("*[Af,Pos,Af-Pos]*").

4.2 The sentence dependency graph

Stanford Dependency Parser (Marneffe et al., 2006) generates, for each sentence, a set of typed dependencies between the words and the following information for each word token: *word, lemma, part of speech, named entity type* (if applicable) and *word index* in the sentence (for order preservation). This output can re-represented as a graph which we call *Dependency Graph* (DG). DG is instantiated from SDP output whose nodes and edge FSs are filled with corresponding information.

4.3 Generating mood parse

The *mood constituency graph* (MCG) is a directed graph which partitions the sentence into constituents at various hierarchical levels. A constituent has one corresponding MCG node. Therefore MCG node FS, among other attributes, contains the list of DG nodes which the constituent covers. The generation of MCG is executed in two phases: *creation* and *enrichment*.

A. *Creation* of MCG is based on breadth-first traversal of DG. The edge type, at every step decides what generative operation to execute on the MCG. The operation choices are: (1) *create a new constituent* (subject, predicator, finite, complement or adjunct as described in section 2.1), (2) *extend the current constituent* by a new token, (3) *create a subordinate clause constituent* and (4) *create a sibling constituent*.

Creation of a new constituent adds a new MCG node under the current one and fills it with the current DG node and all of its children. Extension of constituent means adding the current DG node to MCG node. This is a passive opera-

tion since the current DG node was added already when the new constituent was created. Creation of the clause constituent is similar to the creation of a simple constituent, but additionally, one more clause constituent is added under the former one and they both span over the same DG nodes. Sibling constituent creation adds a new constituent under the parent of the current one. The current DG node and all its children are moved from the current MCG node to a newly created sibling.

dependency relation	*operation on mcg*	*constituent type*
nsubj, nsubjpass, xsubj	new constituent	subject
csubj, csubjpass	new clause constituent	subject
attr, dobj, acomp	new constituent	complement
ccomp	new clause constituent	complement
agent	new constituent	complement agent
iobj	new constituent	complement dative
prep, prepc	new clause constituent	complement or adjunct
advcl	new clause constituent	adjunct
advmod, tmod	new constituent	adjunct
infmod, purpcl, rcmod, ref, rel, parataxis	new constituent	clause
expl, complm, mark	new sibling constituent	Marker
vb-dep-vb, vb-conj-vb,	new constituent	clause
amod, appos, aux, auxpass, cc, det, mwe, neg, nn, npadvmod, num, number, pobj, poss, possessive, preconj, predet, prt, punct, quantmod, xcomp	Extends current constituent	---

Table 4 rules for MCG creation

The decision of what operation to execute is based on the DG edge type, and in a few cases, on edge type plus the word's part of speech. Dependency types that require edge part of speech context are: "*dep*" and "*conj*". lists the rules binding (1) *Stanford Dependency relations*, (2) *generative operation in MCG* and (3) the *constituent type*. The following algorithm outlines how the MCG is created.

```
current_constit = create root node in mcg
bfs traverse DG:
 for each edge:
  oper_type,conit_type = get_rule(edge,nodes)
  new_constit = exec_oper(oper_type,
                contit_type, current_constit)
  current_constit = new_constit
```

B. In the *enrichment* phase Finite and Predicate components are added. Their creation requires more than one edge information available during the DG traversal and therefore, for the simplicity and clearness of the algorithm, these components have been left out of the creation phase. Moreover, in the cases of complex predicates the empty constituents need to be created according to subject/object control rules as described in Section 2.5 in order to constitute full clauses.

Finally, *voice, polarity, mood type and modal deixis features* are added to each clause. For each feature selection in the MOOD system (Section 2.1) a corresponding graph pattern is provided. The algorithm attempted to match these graph patterns in the MCG in order to determine which feature to add to the MCG clause constituent. The following algorithm outlines the enrichment phase of the MCG:

```
for each clause in MCG:
 create finite and predicate constituents
 create empty constituents
 match voice patterns & add features
 match polarity patterns & add features
 match mood type patterns & add features
 match modal deixis patterns & add features
```

4.4 Generating transitivity parse

MCG divides the sentence into clauses and their constituents and so it is an ideal structure to carry transitivity descriptions. Transitivity is a clause-level analysis that decorates the constituents with semantic roles, i.e. the Predicate with Process Type, the Subject and Complements with Participant Roles, the Adjuncts with Circumstances type (not covered here).

Transitivity parsing is very similar to enrichment phase of MCG generation. The following algorithm outlines how to enrich the MCG with transitivity descriptions:

```
for each clause in MCG:
  get process types (main verb)
  for each process type:
    get all configuration GPs
    for each configuration GP:
      if GP matches clause:
        add process type to clause
        extract marked nodes
        add roles to clause constituents
```

The graph patterns used in this task are called *Configuration Graph Patterns (CGP)*. They represent the graph form of the clause configurations as described by Fawcett (2009). Fawcett's configurations are given in a "*normalised*" form. It resembles Chomsky's *kernel sentences* which are of declarative mood type, active voice and unmarked positive polarity. This fixed functional feature set accompanying semantic descriptions

of a configuration yields a particular realisation form. Any alternative feature set yields a predictable alternative realisation that can be grasped by the corresponding Graph Patterns for the same configuration. For example, a variation in voice of a two-role configuration would require two CGPs differing by participant positions. CGP with a passive voice would have switched participant roles between Subject and Complement constituents. So, every configuration may have several realization variations (as a result of conflation with other functions) and each configuration, therefore, has several corresponding CGPs covering those realisation variations.

In the Cardiff Grammar there are 16 distinct process types which cover 65 possible configurations. The process type dictates which configurations are allowed to occur and therefore the process type dictates which set of CGP shall be attempted for matching to clause DG. CGPs are grouped according to the process type and stored in a graph pattern repository.

Transitivity parsing process employs pattern-based node extraction. For each clause in MCG, process types are looked up in PTDB via Main Verb lexeme. Then, for each process type, all CGPs are matched against the clause MCG and in case of a successful match the marked nodes are extracted and enriched with semantic information carried in CGP. The final result is a MCG with a richer feature structure containing functional-semantic information specific for each clause constituent covered by the clause.

5 Conclusions

The present paper describes a graph-based approach to generate SFG mood and transitivity parses from the Stanford Dependency parse and Process Type Database. It is a computationally and linguistically viable text parsing approach for natural language understanding which encompasses framed semantic roles together with an adequate syntactic structure to support those semantic roles.

The presented method relies on correctness of dependency parse produced by SDP and on correctness of entries from PTDB. This constitutes a weak point because errors in SDP or PTDB can lead to decreased overall correctness. In case of missing verb items or verb senses for that verb items the parser will fail to produce transitivity analysis. Or if the verb sense has a faulty configuration specification then it will lead to incorrect semantic labelling. In case of incorrect dependencies or dependency types the mood parsing is likely to be erroneous as well. We cannot tell yet to what extend these limitations influence the correctness of our approach and it constitutes a future work.

A valuable investigation would be to check whether the Semantic Role Labelling with Cardiff Grammar suffers from the same limitation as the approaches describe in CoNLL-2005 which records a dramatic drop in parse correctness when the test corpus differs from the training corpus.

The semantic analysis provided by TRANSITIVITY covers process and participants. Currently no circumstance type has been taken into account as it would require additional lexicogrammatical resources.

No wide coverage parser employing the full Sydney Grammar has yet eventuated. However, the demand for systemic-oriented sentence analysis is on rise. Another increasing demand is for semantic text analysis to further support natural language understanding process. Concurrently there is a pragmatic need to work with unrestricted text and within reasonably small time for offline tasks like information extraction from large documents, and within significantly small time for online tasks like in the case of Dialogue Systems. The current method manages to satisfy demand for systemic sentence analysis via a trade-off between the richness of Sydney Grammar and pragmatic needs regarding coverage and execution time. Even so, a wide coverage systemic parser could have applications well beyond those traditionally explored with automatic semantic and syntactic analysis and become of critical value for solving real-life problems.

Bibliography

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley FrameNet Project. In C. Boitet & P. Whitelock (Eds.), *Proceedings of the 36th annual meeting on Association for Computational Linguistics* (Vol. 1, pp. 86–90). Association for Computational Linguistics.

Bateman, J. A. (2008). Systemic-Functional Linguistics and the Notion of Linguistic Structure: Unanswered Questions, New Possibilities. In Jonathan J. Webster (Ed.), *Meaning in Context: Implementing Intelligent Applications of Language Studies* (pp. 24–58). London, New York: Continuum.

Carpenter, B. (1992). *The logic of typed feature structures*. Cambridge: Cambridge University Press.

Carreras, X., & Màrquez, L. (2005). Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. *Proceedings of the Ninth Conference on Computational Natural Language Learning CoNLL2005*, 152–164.

Cordella, L. P., Foggia, P., Sansone, C., & Vento, M. A (sub)graph isomorphism algorithm for matching large graphs. , 26 IEEE Transactions on Pattern Analysis and Machine Intelligence 1367–72 (2004). IEEE Computer Society.

Fawcett, R. P. (2008). *Invitation to Systemic Functional Linguistics through the Cardiff Grammar*. Equinox Publishing Ltd.

Fawcett, R. P. (2009). How to Analyze Process and Participant Roles. In *The Functional Semantics Handbook: Analyzing English at the level of meaning*. London: Continuum.

Fillmore, C. J. (1985). Frames and the semantics of understanding. *Quaderni di Semantica*, *6*, 222–254.

Haegeman, L. (1991). *Introduction to Government and Binding Theory*. *Blackwell Textbooks in Linguistics 1* (Vol. 2, p. 701). Blackwell.

Halliday, M. A. K., & Matthiessen, C. (2004). *An introduction to functional grammar*. London: Hodder Education.

Honnibal, M., & Curran, J. R. (2005). Creating a Systemic Functional Grammar Corpus from the Penn Treebank. In *Proceedings of the 5th Workshop on Important Unresolved Matters* (pp. 89–96). Association for Computational Linguistics.

Kipper, K., Korhonen, A., Ryant, N., & Palmer, M. (2008). A large-scale classification of English verbs. *Language Resources And Evaluation*, *42*, 21–40.

Klein, D., & Manning, C. (2003). Accurate unlexicalized parsing. (E. Hinrichs & D. Roth, Eds.)*Proceedings of the 41st Annual Meeting on Association for Computational Linguistics ACL 03*, *1*, 423–430.

Marneffe, M.-C., MacCartney, B., & Manning, C. D. (2006). Generating Typed Dependency Parses from Phrase Structure Parses. In *LREC 2006* (Vol. 6, pp. 449–454).

Marneffe, M.-C., & Manning, C. D. (2008). The Stanford typed dependencies representation. (P. Neittaanmäki, T. Rossi, K. Majava, & O. Pironneau, Eds.)*Coling 2008 Proceedings of the workshop on CrossFramework and CrossDomain Parser Evaluation CrossParser 08*, *1*, 1–8.

Martin Kay. (1985). Parsing In Functional Unification Grammar. In D.Dowty, L. Karttunen, & A. Zwicky (Eds.), *Natural Language Parsing*. Cambridge University Press.

Matthiessen, C. (1985). The systemic framework in text generation: Nigel. In James Benson and Willian Greaves (Ed.), *Systemic perspective on Discourse, Vol I* (pp. 96–118). Norwood, New Jersey: Ablex.

Michael O'Donnell. (2012). Transitivity Development in Spanish Learners of English. In *Proceedings of 39th International Systemic Functional Linguistics Conference*. Sydney, Australia.

Minsky, M. (1974). A framework for representing knowledge. In P. Winston (Ed.), *The Psychology of Computer Vision* (Vol. 20, pp. 211–277). McGraw-Hill.

Neale, A. C. (2002). *More Delicate TRANSITIVITY: Extending the PROCESS TYPE for English to include full semantic classifications*. Cardiff.

O'Donnell, M. (1993). Reducing Complexity in Systemic Parser. In *Proceeedings of the Third International Workshop on Parsing Technologies*. Tilburg.

O'Donnell, M. (1994). *Sentence Analysis and Generation: a systemic perspective*. Sydney.

O'Donnell, M. (2005). The UAM Systemic Parser. In *Proceedings of the 1st Computational Systemic Functional Grammar Conference*. Sydney: University of Sydney.

Quirk, R., Greenbaum, S., Leech, G., Svartvik, J., & Crystal, D. (1985). *A comprehensive grammar of the English language*. (R. Quirk, Ed.)*Computational Linguistics* (Vol. 1, p. 1779). New York, New York, USA: Longman.

Robert Kasper. (1988). An Experimental Parser for Systemic Grammars. In *Proceedings of the 12th Int. Conf. on Computational Linguistics*. Budapest.